Exhibit B to

Response to Office Action

Computer Architecture A Quantitative Approach

David A. Patterson
UNIVERSITY OF CALIFORNIA AT BERKELEY

John L. Hennessy
STANFORD UNIVERSITY

With a Contribution by
David Goldberg
Xerox Palo Alto Research Center

MORGAN KAUFMANN PUBLISHERS, INC.
SAN MATEO, CALIFORNIA

Sponsoring Editor Bruce Spatz
Production Manager Shirley Jowell
Technical Writer Walker Cunningham
Text Design Gary Head
Cover Design David Lance Goines
Copy Editor Linda Medoff
Proofreader Paul Medoff
Computer Typesetting and Graphics Fifth Street Computer Services

Library of Congress Cataloging-in-Publication Data

Patterson, David A.
Computer architecture : a quantitative approach / David A. Patterson, John L. Hennessy
p. cm.
Includes bibliographical references
ISBN 1-55880- 069-8
1. Computer architecture. 2. Electronic digital computers --Design and construction. I. Hennessy, John L. II. Title.
QA76.9.A73P377 1990
004.2'2--dc20 89-85227
CIP

Morgan Kaufmann Publishers, Inc.
Editorial Office: 2929 Campus Drive. San Mateo, CA 94403
Order from: P.O. Box 50490, Palo Alto, CA 94303-9953

ADVICE, PRAISE, & ERRORS: Any correspondence related to this publication or intended for the authors should be addressed to the editorial offices of Morgan Kaufmann Publishers, Inc., Dept. P&H APE. Information regarding error sightings is encouraged. Any error sightings that are accepted for correction in subsequent printings will be rewarded by the authors with a payment of $1.00 (U.S.) per correction. Electronic mail can be sent to bugs@vsop.stanford.edu.

INSTRUCTOR SUPPORT: For information on classroom software and other instructor materials available to adopters, please contact the editorial offices of Morgan Kaufmann Publishers, Inc.

significant bit of the address is used to select the appropriate half of the TLB (step 1). Since the system portion of the address space is the same for all processes, a process switch invalidates only the lower 32 entries of each bank for the VAX-11/780 TLB. This restriction had two goals. The first was to reduce the process-switch time by reducing the number of TLB entries that had to be invalidated; the second was to improve performance by preventing the system or user process from throwing out the other's translations when process switches were frequent. Splitting the TLB will usually lead to higher overall TLB miss rate, but may reduce the peak TLB miss rate in heavily process-switching environments.

A Segmented Virtual Memory Example: Protection in the Intel 80286/80386

The second system is the most dangerous system a man ever designs. . . . The general tendency is to over-design the second system, using all the ideas and frills that were cautiously sidetracked on the first one.

F. P. Brooks, Jr., *The Mythical Man-Month* (1975)

The original 8086 used segments for addressing, yet it provided nothing for virtual memory or for protection. Segments had base registers but no bound registers and no access checks; and before a segment register could be loaded the corresponding segment had to be in physical memory. Intel's dedication to virtual memory and protection is evident in subsequent models, with a few fields extended to support larger addresses.

Like the VAX, the 80286 has four levels of protection. The innermost level (0) corresponds to VAX kernel mode, and the outermost level (3) corresponds to VAX user mode. The 80286 also follows the VAX by having separate stacks for each level to avoid security breaches between the levels. There are also data structures analogous to VAX page tables that contain the physical addresses for segments, as well as a list of checks to be made on translated addresses.

The Intel designers did not stop there. The 80286 divides the address space, allowing both the operating system and the user access to the full space. The 80286 user can call an operating system routine in this space and even pass parameters to it retaining full protection. This is not a trivial action, since the stack for the operating system is different from the user's stack. Moreover, the 80286 allows the operating system to maintain the protection level of the called routine for the parameters that are passed to it. This potential loophole in protection is prevented by not allowing the user to ask the operating system to access something indirectly that he would not have been able to access himself. Such security loopholes are called *Trojan horses*.

The 80286 designers were guided by the principle of trusting the operating system as little as possible, while supporting sharing and protection. As an example of the use of such protected sharing, suppose a payroll program writes checks and also updates the year-to-date information on total salary and benefits payments. Thus, we want to give the program the ability to read the salary and

year-to-date information and modify the year-to-date information but not the salary. We shall see the mechanism to support such features shortly. In the rest of this section we will look at the big picture of the 80286 protection and examine its motivation. Readers interested in the detailed picture can find it in a comprehensive book by Crawford and Gelsinger [1987].

Adding Bounds Checking and Memory Mapping

The first step in enhancing the 80286 was getting the segmented addressing to check bounds as well as supply a base. Rather than a base address, as in the 8086, segment registers in the 80286 contain an index to a virtual memory data structure called a *descriptor table*. Descriptor tables play the role of page tables in the VAX. On the 80286 the equivalent of a page-table entry is a *segment descriptor*. It contains fields found in PTEs:

A *present bit*—equivalent to the PTE valid bit, used to indicate this is a valid translation

A *base field*—equivalent to a page-frame address, containing the physical address of the first byte of the segment

An *access bit*—like the reference bit or use bit in some architectures that is helpful for replacement algorithms

An *attributes field*—like the protection field in the VAX PTE, which specifies the valid operations and protection levels for operations that use this segment

There is also a *limit field*, not found in paged systems, which establishes the upper bound of valid offsets for this segment. Figure 8.30 shows examples of 80286 segment descriptors.

Adding Sharing and Protection

The Intel designers' next step was to provide for protected sharing. Like the VAX, half of the address space is shared by all processes and half is unique to each process, called *global address space* and *local address space*, respectively. Each half is given a descriptor table with the appropriate name. A descriptor pointing to a shared segment is placed in the global-descriptor table, while a descriptor for a private segment is placed in the local-descriptor table.

A program loads an 80286 segment register with an index to the table and a bit saying which table it desires. The operation is checked according to the attributes in the descriptor, the physical address being formed by adding the offset in the CPU to the base in the descriptor, provided the offset is less than the limit field. Unlike the encoding of operations and levels in the VAX PTE, every segment descriptor has a separate two-bit field to give the legal access level of this segment. A violation occurs only if the program tries to use a segment with a lower protection level in the segment descriptor.

We can now show how to invoke the payroll program to update the year-to-date information without allowing it to update salaries. The program could be given a descriptor to the information that has the writable field clear, meaning it can read but not write the data. A trusted program can then be supplied that will only write the year-to-date information and is given a descriptor with the writable field set (Figure 8.30). The payroll program invokes the trusted code using a code-segment descriptor with the conforming field set (Figure 8.30). This means the called program takes on the privilege level of the code being called rather than the privilege level of the caller. Hence, the payroll program can read the salaries and call a trusted program to update the year-to-date totals, yet the payroll program cannot modify the salaries. If a Trojan horse exists in this system, to be effective it must be located in the trusted code whose only job is to update the year-to-date information. The argument for this style of protection is that limiting the scope of the vulnerability enhances security.

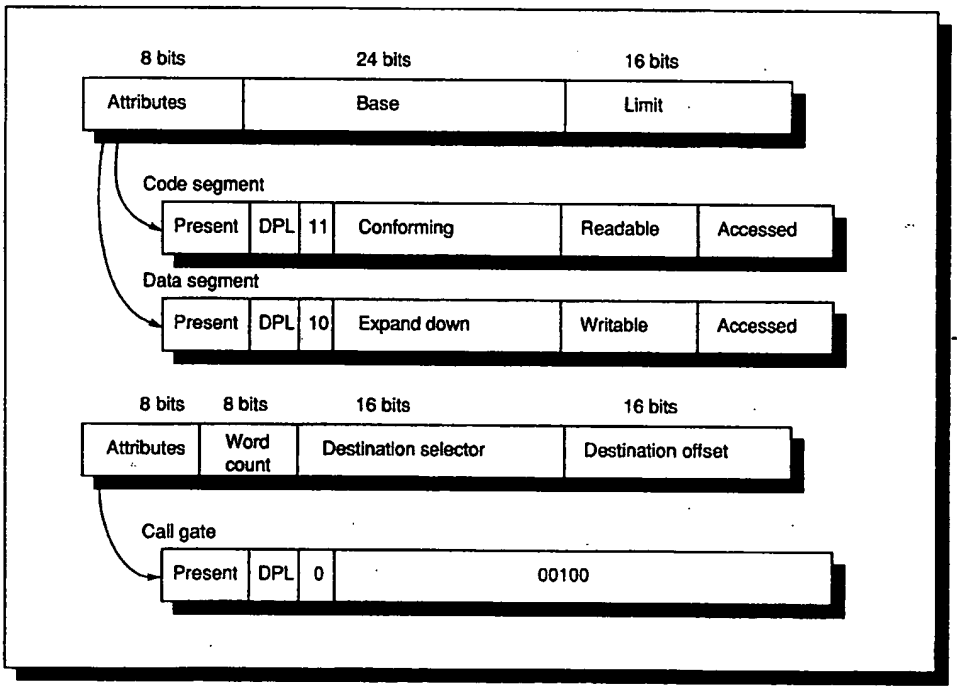

FIGURE 8.30 The 80286 segment descriptors are all 48 bits long and are distinguished by bits in the attributes field. *Base*, *limit*, *present*, *readable*, and *writable* are all self-explanatory. DPL means *descriptor privilege level*—this is checked against the code privilege level to see if the access will be allowed. *Conforming* says the code takes on the privilege level of the code being called rather than the privilege level of the caller; it is used for library routines. The *expand-down field* flips the check to let the base field be the high-water mark and the limit field be the low-water mark. As one might expect, this is used for stack segments that grow down. *Word count* controls the number of words copied from the current stack to the new stack on a call gate. The other two fields of the call-gate descriptor, *destination selector* and *destination offset*, select the descriptor of the destination of the call and the offset into it. There are many more than these three segment descriptors in the 80286. The principal change in the 80386 was to lengthen the base by eight bits and the limit by four bits.